Algebraic Methods for Interactive Proof Systems

Carsten Lund*
Lance Fortnow†
Howard Karloff‡
University of Chicago

Noam Nisan§
Hebrew University

Abstract

We present a new algebraic technique for the construction of interactive proof systems. We use our technique to prove that every language in the polynomial-time hierarchy has an interactive proof system. This technique played a pivotal role in the recent proofs that IP=PSPACE (Shamir) and that MIP=NEXP (Babai, Fortnow and Lund).

1 Introduction

NP can be viewed as the set of languages L with this property: there is a deterministic polynomial-time verifier (Vanna) and an infinitely-powerful prover (Pat) such that for all x, if x is in L, then in polynomial time Pat can persuade Vanna that x is in L, and if x is not in L, then *no* prover, Pat or any other, can persuade Vanna that x *is* in L. Pat and Vanna communicate on a two-way channel (though two-way communication is not necessary here). For example, Pat can convince Vanna that a graph G is 3-colorable by exhibiting a 3-coloring. If G is not 3-colorable, no prover will ever succeed in persuading Vanna that G is 3-colorable.

(Of course, co-NP-complete languages are not thought to be in NP. No prover Pat is known who can convince a skeptical deterministic verifier that a graph is not 3-colorable.)

We can extend this idea of "provability" by allowing Vanna to flip coins and by requiring instead that if x is in L, with probability at least 2/3 Pat persuades Vanna that x is in L, and if x is not in L, only with probability at most 1/3 can any prover persuade Vanna that x is in L. Babai [B] and Goldwasser, Micali and Rackoff [GMR] developed this *interactive proof system* model. A summary of previous results on interactive proof systems can be found in [BM].

While certain problems such as graph non-isomorphism which are not known to be in NP were known to have interactive proofs [GMW], computer scientists generally believed the class IP of languages accepted by interactive proofs not much larger than NP. It was believed that co-NP-complete languages did not have interactive proofs.

We prove that interactive proofs systems have far greater power than originally believed. Our main result is an interactive proof for the language $\{(A,s)|s$ is the permanent of 0,1 matrix $A\}$. When combined with the fact that the permanent of 0,1 matrices is #P-complete [V] and the fact that #P is hard for the polynomial-time hierarchy [T], the existence of an interactive proof for the permanent implies that every language in the polynomial-time hierarchy has an in-

*Supported by a fellowship from the University of Århus.
†Supported by NSF grant CCR-9009936.
‡Supported by NSF grant CCR-8807534.
§Some of this work was performed at MIT and supported by NSF grant CCR-865727 and ARO grant DLL03-86-K-017.

CH2925-6/90/0000/0002$01.00 © 1990 IEEE

teractive proof. In particular this means that every language in co-NP has an interactive proof, even the complement of 3-COLORABILITY, for example.

For the proof we develop a new technique for reducing the problem of verifying the value of a low-degree polynomial at two points to verifying the value at one new point. Shamir [Sh] has extended our technique to show that all languages in PSPACE have interactive proofs. From the fact that IP⊆PSPACE [Pa], it follows that IP=PSPACE. Babai, Fortnow and Lund [BFL] have extended this technique in a different direction to prove that every language in nondeterministic exponential time has a two-prover interactive proof in which the provers cannot communicate with each other.

Our results also have implications for program checking, verification and self-correction in the context of Blum and Kannan [BK], Blum, Luby and Rubinfeld [BLR] and Lipton [Li] (which uses ideas of Beaver and Feigenbaum [BeF]). In fact, the Blum-Luby-Rubinfeld and Lipton papers inspired our result.

Our result does not relativize. Fortnow and Sipser [FS] have created an oracle under which co-NP does not have interactive proofs. To our knowledge this is the first result to "go contrary" to a previously-published oracle. Subsequent to the appearance of our paper, Chor, Goldreich and Hastad [CGH] proved the same relativized result for a random oracle.

2 The Protocol

Theorem 1. Every language in $\mathrm{BPP}^{\#\mathrm{P}}$ has an interactive proof system.

Together with Toda's result that $\mathrm{P}^{\#\mathrm{P}}$ contains all the languages of the polynomial-time hierarchy [T], Theorem 1 implies

Corollary 2. Every language in the polynomial-time hierarchy has an interactive proof system. In particular, every language in co-NP has an interactive proof.

We list some crucial facts about the permanent of a matrix A. If $A = (a_{ij})$ is $r \times r$, the *permanent* $\mathrm{per}(A) = \sum_\sigma a_{1\sigma(1)}a_{2\sigma(2)}\cdots a_{r\sigma(r)}$, where the sum is over all permutations σ of $\{1,2,...,r\}$. We can equivalently define the permanent recursively as $\mathrm{per}(A) = \sum_{1\le i\le r} a_{1i}\cdot \mathrm{per}(A_{1|i})$ where $A_{1|i}$, the $(1,i)$-minor of A, is the matrix A without the first row and the ith column. The number of perfect matchings in an N-boy, N-girl bipartite graph G is equal to the permanent of G's adjacency matrix.

We will exhibit an interactive proof for verifying the permanent of a 0,1 matrix. From the fact that computing the permanent of 0,1 matrices is #P-complete [V], we can reduce any language in $\mathrm{BPP}^{\#\mathrm{P}}$ to a series of questions about the permanents of certain 0,1 matrices. Theorem 1 will follow.

Throughout most of this paper we will work with the permanent over $\mathbb{Z}_p$ of an $N \times N$ matrix A with entries in $\mathbb{Z}_p$ (where p is a prime in the range $(N!, 2^{N^2})$). If A is 0,1, then the permanent of A over $\mathbb{Z}_p$ coincides with its permanent as an integer matrix, since the permanent of an $N \times N$ 0,1 matrix cannot exceed $N!$. We use the crucial fact that for $r \le N$, if B is an $r \times r$ matrix over $\mathbb{Z}_p$ whose entries are linear polynomials over $\mathbb{Z}_p$, then $\mathrm{per}(B)$ is a polynomial of degree at most r over $\mathbb{Z}_p$. Compared to p, r is minuscule.

We will use this fact to "trip up" a cheating prover. We will ask him for the function $f(x) = \mathrm{per}(C + x(D - C))$, where C, D are $r \times r$ matrices over $\mathbb{Z}_p$. The function f is a polynomial of degree at most r. If Pat gives us instead a different polynomial $g(x)$ of degree at most r, then we know that $f(x)$ and $g(x)$ coincide on at most r points, since two polynomials of degree at most r over $\mathbb{Z}_p$ that coincide on $r+1$ points are identical. This fact will be used to ensure that if Pat lies about a low degree polynomial, then with high probability he must lie at a randomly chosen point. Eventually Pat's lie will be discovered. This algebraic technique, used in Lemma 3, also forms an important part of the protocols of Shamir [Sh] and Babai, Fortnow and Lund [BFL].

The verifier Vanna will maintain a list $\mathcal{L}$ of pairs $\mathcal{L} =< (B_1, q_1), (B_2, q_2), ..., (B_t, q_t) >$, where B_i is a square matrix and $q_i \in \mathbb{Z}_p$. All the matrices in $\mathcal{L}$ are of the same size. Initially $\mathcal{L} =< (A, s) >$. If

$s = \text{per}(A)$, then a prover who truthfully answers all of Vanna's questions will induce Vanna eventually to shrink the list to a single pair (B, q), where B is 1×1 and $q = \text{per}(B)$. At that point Vanna will correctly accept the input.

If $s \neq \text{per}(A)$, however the prover answers Vanna's questions, with very high probability the list will *always* contain at least one pair (B_i, q_i) such that $q_i \neq \text{per}(B_i)$. When the list shrinks to one pair (B, q) where B is 1×1 and $q \neq \text{per}(B)$, Vanna will reject the input (if not earlier).

How Vanna manipulates the list is the crux of the protocol. When $\mathcal{L} =< (B, q) >$ ($B = (b_{ij})$, $1 \leq i, j \leq r$, and $r > 1$), for each $i = 1, 2, ..., r$ Vanna constructs the minor $B_i = B_{1|i}$, asks Pat for the permanent of B_i, and gets q_i in return. Vanna checks that $q = \sum_{i=1}^{r} b_{1i} q_i$; if not, she halts and rejects. If $q = \sum_{i=1}^{r} b_{1i} q_i$, she expands $\mathcal{L}$ by replacing $\mathcal{L}$ by $<(B_1, q_1), (B_2, q_2), ..., (B_r, q_r)>$. Provided that $q \neq \text{per}(B)$, $q_i \neq \text{per}(B_i)$ for some i.

When the list has more than one pair, Vanna shrinks the list by replacing the first two pairs $(C, c), (D, d)$ by a new pair (E, e), in the following way. The function $f(x) = \text{per}(C + x(D - C))$ is a polynomial of degree at most r over $\mathbb{Z}_p$. Vanna asks Pat for the $r + 1$ coefficients of f and constructs a polynomial g from the responses. (Or Vanna could just ask for the value of f at $r + 1$ arbitrary points and interpolate herself.) If $g(0) \neq c$ or $g(1) \neq d$, Vanna rejects.

Vanna now uniformly chooses a random $a \in \mathbb{Z}_p$, sends it to Pat, constructs $E = C + a(D - C)$ and $e = g(a)$, and replaces the pairs $(C, c), (D, d)$ in $\mathcal{L}$ by the one pair (E, e). The crucial fact is that if $c \neq \text{per}(C)$ or $d \neq \text{per}(D)$, then with probability at least $1 - r/p$, $\text{per}(E) \neq e$. This follows from Lemma 3.

Lemma 3. Let C and D be $r \times r$ matrices over $\mathbb{Z}_p$. Let g be a polynomial of degree at most r over $\mathbb{Z}_p$ such that either $g(0) \neq \text{per}(C)$ or $g(1) \neq \text{per}(D)$. Then if a is chosen uniformly at random from $\mathbb{Z}_p$,

$$Pr[\text{per}(C + a(D - C)) = g(a)] \leq \frac{r}{p}.$$

Proof. Let $f(x) = \text{per}(C + x(D - C))$, a polynomial of degree at most r over $\mathbb{Z}_p$. Since $f(0) = \text{per}(C)$ and $f(1) = \text{per}(D)$, clearly $f \neq g$. But two nonidentical polynomials of degree at most r over $\mathbb{Z}_p$ can coincide on at most r points. It follows that there are at most r values a such that

$$g(a) = f(a) = \text{per}(C + a(D - C)). \blacksquare$$

If $\mathcal{L} =<(B_1, q_1), (B_2, q_2), ..., (B_t, q_t)>$ and at least one i satisfies $\text{per}(B_i) \neq q_i$, then after $t - 1$ shrinkings $\mathcal{L}$ will consist of one pair (H, h) with $h \neq \text{per}(H)$ with very high probability. The idea, then, is to replace the initial list $\mathcal{L} =<(A, s)>$ by lists of smaller and smaller matrices, until eventually $\mathcal{L} =<(B, q)>$ where B is 1×1. If $q \neq \text{per}(B)$—a condition Vanna can easily test—Vanna will reject. Otherwise she'll accept.

How likely is it that Vanna will be able to maintain the "invariant?" A sequence of one expansion step followed by $r - 1$ shrinking steps will replace $\mathcal{L} =<(B, q)>$, where B is $r \times r$, by $\mathcal{L} =<(B', q')>$, where B' is $(r - 1) \times (r - 1)$. Thus fewer than N^2 steps (of either kind) suffice to reduce $\mathcal{L} =<(A, s)>$ to $\mathcal{L} =<(B, q)>$, where B is 1×1. It follows that the probability that a cheating prover can induce Vanna to erroneously accept (A, s) is less than N^2 times the minuscule probability (at most N/p) that a given expand or shrink step first violates the "invariant."

Here is Vanna's protocol in detail. A is an $N \times N$ 0,1 matrix, $0 \leq s \leq N!$.

begin

Let $\mathcal{L} =<(A, s)>$. Pat picks an integer p in $(N!, 2^{N^2})$ and provides a short proof to Vanna that p is prime [Pr]. (Or Vanna can generate a prime in $(N!, 2^{N^2})$ via a randomized primality algorithm.) All arithmetic in this protocol is done mod p.

Repeat until $\mathcal{L} =<(B, q)>$ for some 1×1 B:

If $\mathcal{L} =<(B, q)>$ and $B = (b_{ij})$ is $r \times r$, then do

Expand: Vanna defines $B_i = B_{1|i}$. Vanna asks Pat for the permanents of B_i, $1 \leq i \leq r$, and gets q_i for the permanent of B_i. If $\sum_{i=1}^{r} b_{1i} q_i \neq q$, Vanna rejects. Otherwise, she sets $\mathcal{L} = <(B_1, q_1), (B_2, q_2), ..., (B_r, q_r)>$.

Else ($\mathcal{L}$ has two or more pairs) do

Shrink: Vanna chooses the first two pairs (C, c) and (D, d) from $\mathcal{L}$, asks Pat for the $r+1$ coefficients of $f(x) = \text{per}(C + x(D - C))$, and constructs $g(x)$ from them. If $g(0) \neq c$ or $g(1) \neq d$, Vanna rejects. Otherwise, she chooses a random $a \in \mathbb{Z}_p$, sends it to Pat and replaces the pairs $(C, c), (D, d)$ in $\mathcal{L}$ by $(C + a(D - C), g(a))$.

When $\mathcal{L} = <(B, q)>$ and B is 1×1, Vanna accepts if $q = \text{per}(B)$ or rejects if $q \neq \text{per}(B)$.

end.

The protocol runs for exactly $N + (N-1) + (N-2) + \cdots + 2$ iterations if Vanna accepts (A, s).

Theorem 4.

(1) There is a prover Pat such that for all $N \times N$ 0,1 matrices A and $0 \leq s \leq N!$, if $\text{per}(A) = s$ then $Pr[\text{Vanna accepts } (A, s)] = 1$.

(2) For all (A, s), if $\text{per}(A) \neq s$, then for all provers, $Pr[\text{Vanna accepts } (A, s)] < N^3/p$.

Proof.

(1) If Pat truthfully answers all of Pat's questions, it is easy to see that with probability one eventually $\mathcal{L} = <(B, q)>$ with B 1×1 and $q = \text{per}(B)$. At this point Vanna accepts.

(2) Suppose $\text{per}(A) \neq s$. Fix a prover Pat. If Vanna accepts (A, s), then at some time $\mathcal{L} = <(B, q)>$ with $q = \text{per}(B)$; initially $\mathcal{L} = <(A, s)>$ with $s \neq \text{per}(A)$. Say Pat *succeeds in iteration* m if Vanna runs the protocol for at least m iterations, and

$$q = \text{per}(B) \text{ for all } (B, q) \text{ in } \mathcal{L}$$

first occurs just after the repeat loop has been executed m times. Pat succeeds in some iteration if Vanna accepts. Since there are only $N + (N-1) + \cdots + 2 < N^2$ iterations, it suffices to prove that $Pr[\text{Pat succeeds in iteration } m] < N/p$.

Fix an m. Without loss of generality, we may assume that Pat never induces Vanna to reject until $\mathcal{L}$ consists of only one pair, the matrix of which is 1×1. (Otherwise, we may replace Pat by another prover Pat' who, instead of inducing Vanna to reject early, answers the remaining questions in a way consistent with his earlier responses, as long as possible. Against such a prover Vanna will not halt until the last stage of the protocol. The probability that Pat succeeds in iteration m is no greater than the probability that Pat' does.)

Pat simply cannot succeed in an *Expand* step: if $\mathcal{L} = <(B, q)>$ with $q \neq \text{per}(B)$ becomes $\mathcal{L} = <(B_1, q_1), ..., (B_r, q_r)>$ with $q_i = \text{per}(B_i)$ for all i, Vanna immediately rejects.

If iteration m is a *Shrink* step, $\mathcal{L}$ contains (C, c) and (D, d) before the step and (E, e) afterward. If $c = \text{per}(C)$ and $d = \text{per}(D)$, then $\mathcal{L}$ contained a pair (H, h) with $h \neq \text{per}(H)$. It still does. So we may assume that either $c \neq \text{per}(C)$ or $d \neq \text{per}(D)$. In this case Lemma 3 tells us that $Pr[e = \text{per}(E)] < N/p$. Thus

$$Pr[\text{Pat succeeds in iteration } m] < \frac{N}{p}. \blacksquare$$

3 Extensions

The protocol above requires $\Omega(N^2)$ rounds of prover-verifier communication when the input matrix is $N \times N$. Babai has suggested the following scheme to reduce the number of rounds to $O(N)$. His idea makes it possible to shrink a list $\mathcal{L}$ with r pairs into a list $\mathcal{L}'$ with one pair in one round. For $1 \leq i \leq r$, let

$$f_i(x) = \prod_{j : j \neq i, 1 \leq j \leq r} \frac{(x - j)}{(i - j)}.$$

Note that $f_i(x)$ is a polynomial of degree $r-1$ with $f_i(i) = 1$, and if $j \neq i$, then $f_i(j) = 0$ for all j, $1 \leq j \leq r$. Let $\mathcal{L} = <(B_1, q_1), (B_2, q_2), ..., (B_r, q_r)>$ and define $C(x) = \sum_{i=1}^{r} f_i(x) B_i$. The matrix $C(x)$ has entries consisting of polynomials of degree at most $r-1$. Now $f(x) = \text{per}(C(x))$ is a polynomial of degree at most $r(r-1)$ with $f(i) = \text{per}(C(i)) = \text{per}(B_i)$ for all i, $1 \leq i \leq r$. This gives us an alternative *Shrink* procedure:

Shrink: Vanna asks Pat for the $r(r-1)+1$ coefficients of $f(x) = \text{per}(C(x))$, and constructs $g(x)$ from them. If $g(i) \neq q_i$ for some $1 \leq i \leq r$, then Vanna rejects. Otherwise, she chooses a random $a \in \mathbb{Z}_p$, sends it to Pat and replaces $\mathcal{L}$ by $<(C(a), g(a))>$.

The proof of correctness is similar to the proof of Theorem 4 and is omitted here.

Because the number of rounds of an interactive proof system can be reduced by a constant factor [BM], for any $\epsilon > 0$ there is a variant of our permanent protocol running in at most ϵN rounds. A bounded-round protocol for the permanent would imply that the polynomial-time hierarchy collapses, since Boppana, Hastad and Zachos [BHZ] showed that if all co-NP languages have bounded-round protocols, then the hierarchy collapses. To our knowledge this is the first example of an interactive proof system that appears to require an unbounded number of rounds.

Babai and Fortnow [BF] and Shamir [Sh] have provided alternate interactive proofs for verifying the values of #P functions by counting the number of assignments satisfying a CNF formula, thus circumventing the need for Valiant's result on the completeness of the permanent. Given a 3CNF formula F and a value q we will describe an interactive proof for verifying that q is the number of satisfying assignments to F. For example, suppose that F has m clauses and

$$F = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_4 \vee \neg x_1 \vee \neg x_2) \cdots$$

We convert F to an arithmetic formula $f(x_1, \ldots, x_n)$ such that, where one represents true and zero, false, $f(x_1, \ldots, x_n)$ equals the truth value of F on assignment $(x_1, \ldots, x_n)$. First we convert each clause to a polynomial whose value is one if the clause is satisfied and zero otherwise. For example we transform the clause $(x_1 \vee x_2 \vee \neg x_3)$ into

$$1 - [(1 - x_1)(1 - x_2) \cdot x_3].$$

We then let f be the product of all these polynomials.

Now notice that the expression

$$\sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n)$$

is exactly the number of assignments that satisfy F. For each i, $0 \leq i \leq n$, let

$$f_i(x_1, \ldots, x_i) = \sum_{x_{i+1}=0}^{1} \sum_{x_{i+2}=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, \ldots, x_n).$$

Note that f_0 is the number of assignments that satisfy F, $f_{i-1}(x_1, \ldots, x_{i-1}) = \sum_{x_i=0}^{1} f_i(x_1, \ldots, x_i)$, f_n is computable in polynomial time and each f_i is an i-variable polynomial of degree at most $3m$.

The protocol is very similar to that for the permanent. First Pat chooses a prime p, $2^n < p < 2^{2n}$, and gives a short proof that p is prime to Vanna. Then Pat gives Vanna the value g_0 (supposedly equal to f_0) and a polynomial $g_1(x)$ that Pat claims equals $f_1(x)$. Vanna verifies that $g_1(0) + g_1(1) = g_0$ and chooses a random $a_1 \in \mathbb{Z}_p$. Pat then gives Vanna $g_2(x)$ which purportedly equals $f_2(a_1, x)$ and Vanna once again verifies the sum and chooses a_2. They repeat this process for each f_i and at the end Vanna verifies the value $f_n(a_1, \ldots, a_n)$. The proof of correctness is virtually identical to that for the permanent.

We would like to arithmetize QBF formulas in a way that allows a similar interactive proof. If we just replace the $\exists x_i$ by $\sum_{x_i=0}^{1}$ and the $\forall x_i$ by $\prod_{x_i=0}^{1}$ then the f_i may have exponential degree. Shamir [Sh] used the following idea to circumvent this problem. We add dummy variables so that the number of $\forall$'s between a variable's "creation" and its final usage is bounded. After each $\forall$, we add "$\exists x'(x' \iff x) \wedge$" for each x "created" between that $\forall$ and the previous one; henceforth, we use x' in place of x. This guarantees that no variable has a "lifespan" of more than one $\forall$ from its creation to its final usage.

For example, suppose that the formula is

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \; F(x_1, x_2, x_3, x_4).$$

Adding additional variables, we transform this to

$$\exists x_1$$

$$\forall x_2$$

$$\exists y_1 (y_1 \iff x_1) \wedge$$

$$\exists x_3$$

$$\forall x_4$$

$$\exists z_1 (z_1 \iff x_2) \wedge \exists z_2 (z_2 \iff y_1) \wedge \exists z_3 (z_3 \iff x_3) \wedge$$

$$F(z_2, z_1, z_3, x_4).$$

Where $(x \iff y)$ below is short for $(xy+(1-x)(1-y))$, we then arithmetize the formula as

$$\sum_{x_1=0}^{1}$$

$$\prod_{x_2=0}^{1}$$

$$\sum_{y_1=0}^{1}(y_1 \iff x_1)\cdot$$

$$\sum_{x_3=0}^{1}$$

$$\prod_{x_4=0}^{1}$$

$$\sum_{z_1=0}^{1}(z_1 \iff x_2)\cdot \sum_{z_2=0}^{1}(z_2 \iff y_1)\cdot \sum_{z_3=0}^{1}(z_3 \iff x_3)\cdot$$

$$f(z_2, z_1, z_3, x_4).$$

The value of this expression is non-zero if and only if the original formula is true. Shamir's technique guarantees that the intermediate polynomials have low degree. The final value may have an exponential number of bits, but for any true formula Pat can always find a prime p with a polynomial number of bits such that the final value mod p is non-zero.

Babai, Fortnow and Lund [BFL] have applied the techniques of this paper to multiple-prover interactive proof systems, defined by Ben-Or, Goldwasser, Kilian and Wigderson [BGKW] as interactive proof systems having a polynomial number of provers unable to communicate among themselves or to see the conversation between any other prover and the verifier. Babai, Fortnow and Lund have shown that any language computable in nondeterministic exponential time has a multiple-prover interactive proof system. Their proof uses ideas similar to those of [BF] and [Sh] in order to reduce the problem to testing the multilinearity of a function. The reduction starts with a nondeterministic exponential-time version of the Cook-Levin [Co] NP-completeness theorem, inspired by Simon [Si]. The resulting 3-CNF formula has exponentially many variables and clauses but clause C_i can be computed in polynomial time from i and the input x. Next, one turns this polynomial-time algorithm into a 3-SAT instance again via Cook-Levin [Co]. An arithmetic encoding similar to [BF] and [Sh] turns the Boolean formulas into polynomials; however, the number of variables is exponential. The provers must select an assignment A satisfying the encoded 3-CNF formula. It is known [FRS] that a multiple prover interactive proof system can simulate an interactive proof system where the prover is an oracle A, i.e., the prover's later answers cannot depend on earlier questions. The next task is to test the correctness of this oracle. This is done by following the protocols above under the assumption that the provers extend the domain of A to a large interval and make this new function multilinear. (We note that every Boolean function $A : \{0,1\}^m \to \{0,1\}$ has a unique extension to a multilinear function $\mathbb{Z}^n \to \mathbb{Z}$.)

The multilinearity test tests whether a function A in n variables over $\mathbb{Z}$, given as an oracle, is multilinear over a large interval. This polynomial-time statistical test rejects if it finds evidence that A is not multilinear; and if it accepts, it "guarantees" (with high confidence) that A agrees with some multilinear function f on a $(1-n^{-c})$-fraction of a large interval. The test works by repeating the following several times: pick $(x_1, \ldots, x_n)$ and i, $1 \leq i \leq n$, at random, then check if $h(y) = A(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$ coincides with a linear function at several y chosen at random.

Cai [Ca] has used the protocols of this paper and of Shamir [Sh] to show that every PSPACE language has a bounded-round multiple-prover interactive proof system.

Furthermore, Fortnow and Lund [FL] have extended the techniques from this and Shamir's [Sh] paper to exhibit a polynomial equivalence between time-space complexity of alternating Turing machines and the time-space complexity of the verifier in a public-coin interactive proof system. In particular, they prove that every language in NC has a interactive proof system with a public-coin, polynomial-time, logarithmic-space verifier.

4 Implications

Goldwasser and Sipser [GS] have shown that one can convert any interactive proof to a new interactive proof where the verifier uses public coins, i.e., the verifier can only flip coins "in front of" the prover. Goldreich, Mansour and Sipser [GMS] have shown how to modify an interactive proof system so that for true instances the verifier is convinced with probability one. Both of these properties already hold for our protocol.

Some simple corollaries that follow from these results:

Corollary 5. If cryptographic one-way functions exist then every language in the polynomial-time hierarchy has a zero-knowledge interactive proof.

Proof. Every language with an interactive proof has a zero-knowledge interactive proof if one-way functions exist [BGGHKMR, IY]. ■

From Shamir [Sh], we infer that all languages computable in polynomial space have zero-knowledge interactive proof systems if cryptographic one-way functions exist.

Corollary 6. If every language in IP has a bounded-round interactive proof, then the polynomial-time hierarchy collapses.

This is immediate from Boppana, Hastad and Zachos [BHZ]. Previously Aiello, Goldwasser and Hastad [AGH] constructed an oracle relative to which the class of languages with unbounded-round interactive proofs differs from those with bounded-round interactive proofs.

Our theorem also has applications to program checking, verification and self-correction. Lipton [Li], using ideas of Beaver and Feigenbaum [BeF], showed that the permanent function has a self-testing/correcting pair, i.e., a pair of functions one of which will verify that a program computes the permanent correctly on most inputs and the other of which converts a program that passes the first test into one that correctly computes the permanent on all inputs with high probability.

Theorem 1 provides a program correctness checker for the permanent:

Corollary 7. There exists a probabilistic polynomial-time machine M that given access to a program P and a matrix A, will output with a high degree of confidence either "P outputs the correct value of the permanent of A" or "P does not correctly compute the permanent of some matrix."

Proof. In the proof of Theorem 1 the prover need only answer questions about the permanents of various matrices. We can have M simulate the verifier and use P as the prover. ■

A further discussion of the relationship between interactive proof systems and program testing can be found in [BFL].

MA is the class of languages accepted by an interactive proof system consisting of a single message from the prover to the verifier followed by probabilistic verification by the verifier. We can think of this class as the set of "publishable proofs," "proofs" that can be written down now and randomly verified years later without any help from the prover. Babai has proven that MA$\subseteq \Sigma_2^P \cap \Pi_2^P$ [B]. Corollary 8 implies that if #P has polynomial-size circuits, then $\mathrm{BPP}^{\#\mathrm{P}}$, and hence the polynomial-time hierarchy, lies in MA$\subseteq \Sigma_2^P \cap \Pi_2^P$.

Corollary 8. If #P has polynomial-size circuits then $\mathrm{BPP}^{\#\mathrm{P}} \subseteq$ MA.

Proof. The prover gives the verifier a circuit computing the permanent. She uses this circuit as the prover in the protocol in Section 2. ■

A general discussion of Corollary 8 appears in [BFL] where it is shown that a similar result holds for PSPACE and EXP. Contrast Corollary 8 with the result of Karp and Lipton [KL] that if NP has polynomial-size circuits, then the polynomial-time hierarchy collapses to Σ_2^P.

5 Further Research

We have shown that every language reducible to a #P-complete problem has an interactive proof, and thus

every language in the polynomial-time hierarchy has an interactive proof. In particular, every language in co-NP has an interactive proof. However, even for a co-NP-complete language, the prover must answer #P-complete questions. Is there an interactive proof system for co-SAT where the prover need only answer questions about the satisfiability of CNF formulas? Such a proof system would give an instance checker for NP-complete languages.

We believe that one should study why this result does not relativize. One simple answer is that we have exhibited an interactive proof for a very specific #P-complete function—the permanent—which is not #P-complete relative to any sufficiently complex oracle (since the permanent does not depend on the oracle). Babai and Fortnow [BF] have exhibited a simple characterization of #P functions by polynomials and have used this characterization to prove the main theorem of this paper without any reference to permanents, and also to simplify the proof of Toda's theorem [T]. This algebraic formulation of #P does not hold in relativized worlds. Studying the algebraic structure of well-known complexity classes may lead to yet more exciting relationships among them.

References

[AGH] Aiello, W., S. Goldwasser and J. Hastad, "On the power of Interaction," *Combinatorica*, to appear. Extended abstract appeared in *Proc. 27th FOCS*, 1986, pp. 368-379.

[B] Babai, L., "Trading Group Theory for Randomness," *Proc. 17th STOC*, 1985, pp. 421-429.

[BF] Babai, L. and L. Fortnow, "A characterization of #P by straight-line programs of polynomials," *Proc. 31st FOCS*, 1990.

[BFL] Babai, L., L. Fortnow and C. Lund, "Non-deterministic Exponential Time has Two-Prover Interactive Protocols," *Proc. 31st FOCS*, 1990.

[BM] Babai, L. and S. Moran, "Arthur-Merlin games: A Randomized Proof System, and a Hierarchy of Complexity Classes," *JCSS* **36** 2, 1988, pp. 254-276.

[BeF] Beaver, D. and J. Feigenbaum, "Hiding Instances in Multioracle Queries," *Proc. 7th Symp. on Theoretical Aspects of Comp. Sci.* , *LNCS* **415**, 1990, pp. 37-48.

[BGGHKMR] Ben-Or, M., O. Goldreich, S. Goldwasser, J. Hastad, J. Kilian, S. Micali and P. Rogaway, "Everything Provable is Provable in Zero-Knowledge," *Proc. Crypto*, 1988.

[BGKW] Ben-Or, M., S. Goldwasser, J. Kilian and A. Wigderson, "Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions," *Proc. 20th STOC*, 1988, pp. 113-131.

[BK] Blum, M. and S. Kannan, "Designing Programs that Check Their Work," *Proc. 21st STOC*, 1989, pp. 86-97.

[BLR] Blum, M., M. Luby and R. Rubinfeld, "Self-Testing and Self-Correcting Programs, with Applications to Numerical Programs," *Proc. 22nd STOC*, 1990, pp. 73-83.

[BHZ] Boppana, R., J. Hastad and S. Zachos, "Does co-NP Have Short Interactive Proofs?", *Information Processing Letters*, **25** 2, 1987, pp. 127-132.

[Ca] Cai, J., "PSPACE is Provable by Two Provers in One Round," manuscript, Princeton University, 1990.

[CGH] Chor, B., O. Goldreich and J. Håstad, "The Random Oracle Hypothesis is False," manuscript, Technion, Haifa, Israel, 1990.

[Co] Cook, S., "The Complexity of Theorem-Proving Procedures," *Proc. 3rd STOC*, 1971, pp. 151-158.

[FL] Fortnow, L. and C. Lund, "Interactive Proof Systems and Alternating Time-Space Complexity," Technical Report 90-11. Computer Science Department. University of Chicago.

[FRS] Fortnow, L., J. Rompel and M. Sipser, "On the Power of Multi-Prover Interactive Protocols," *Proc. 3rd Structure in Complexity Theory Conference*, 1988, pp. 156-161.

[FS] Fortnow, L. and M. Sipser, "Are There Interactive Protocols for co-NP Languages?", *Information Processing Letters* **28**, North-Holland, 1988, pp. 249-251.

[GMW] Goldreich, O., S. Micali and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design," *Proc. 27th FOCS*, 1986, pp. 174-187.

[GMR] Goldwasser, S., S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems," *SIAM Journal on Computing* **18** 1, 1989, pp. 186-208. Extended abstract appeared in *Proc. 17th STOC*, 1985, pp. 291-304.

[GMS] Goldreich, O., Y. Mansour and M. Sipser, "Interactive Proof Systems: Provers that Never Fail and Random Selection," *Proc. 28th FOCS*, 1987, pp. 449-461.

[GS] Goldwasser, S. and M. Sipser, "Private Coins versus Public Coins in Interactive Proof Systems." In S. Micali, editor, *Randomness and Computation*, Volume 5 of *Advances in Computing Research*, JAI Press, 1989, pp. 73-90.

[IY] Impagliazzo, R. and M. Yung, "Direct Minium-Knowledge Computation," *Proc. Crypto*, 1987, LNCS 293, pp. 40-51.

[KL] Karp, R. and R. Lipton, "Some Connections between Nonuniform and Uniform Complexity Classes," *Proc. 12th STOC*, 1980, pp. 302-309.

[Li] Lipton, R., "New Directions in Testing," *Proceedings of the DIMACS Workshop on Distributed Computing and Cryptography*, 1989, to appear.

[Pa] Papadimitriou, C., "Games against Nature," *Proc. 24th FOCS*, 1983, pp. 446-450.

[Pr] Pratt, V., "Every Prime has a Succinct Certificate," *SIAM J. Comp* **4**, 1975, 214-220.

[Sh] Shamir, A., "IP=PSPACE," *Proc. 31st FOCS*, 1990.

[Si] Simon, J. "On Some Central Problems in Computational Complexity," Ph.D. Thesis, Cornell University, Computer Science, Tech Report TR 75-224, 1975.

[T] Toda, S., "On the Computational Power of PP and $\oplus$P," *Proc. 30th FOCS*, 1989, pp. 514-519.

[V] Valiant, L, "The Complexity of Computing the Permanent," *Theoretical Computer Science* **8**, 1979, pp. 189-201.